Monocular Surface Reconstruction using 3D Deformable Part Models

Stefan Kinauer, Maxim Berman, Iasonas Kokkinos

Monocular Surface Reconstruction using 3D Deformable Part Models

✉ Stefan Kinauer*, Maxim Berman*, Iasonas Kokkinos

Center for Visual Computing
CentraleSupélec, Inria
Université Paris-Saclay
{stefan.kinauer, maxim.berman, iasonas.kokkinos}@centralesupelec.fr

Abstract. Our goal in this work is to recover an estimate of an object's surface from a single image. We address this severely ill-posed problem by employing a discriminatively-trained graphical model: we incorporate prior information about the 3D shape of an object category in terms of pairwise terms among parts, while using powerful CNN features to construct unary terms that dictate the part placement in the image. Our contributions are three-fold: firstly, we extend the Deformable Part Model (DPM) paradigm to operate in a three-dimensional pose space that encodes the putative real-world coordinates of object parts. Secondly, we use branch-and-bound to perform efficient inference with DPMs, resulting in accelerations by two orders of magnitude over linear-time algorithms. Thirdly, we use Structured SVM training to properly penalize deviations between the model predictions and the 3D ground truth information during learning.
Our inference requires a fraction of a second at test time and our results outperform those published recently in [17] on the PASCAL 3D+ dataset.

1 Introduction

The advent of deep learning has led to dramatic progress in object detection [11,12] and also in tasks that can lead to 3D object perception, such as viewpoint estimation [25]. Even though Convolutional Networks seem to be the method of choice for such problems, they may not be yet appropriate for structured prediction tasks that are 'beyond detection' and involve multiple, real-valued and interrelated variables that need to be estimated with high precision. The scarcity of data for many of these tasks, as well as the rich structure inherent in the problems advocate a more explicit, modelling-based method.

Our work addresses in particular the problem of estimating the three-dimensional shape of an object from a single RGB image. Apart from the inherent interest of the problem, this can lead to applications in graphics (rendering, augmented reality), robotics (grasping, navigation) and object detection (dataset augmentation, 3D-based classification). This severely ill-posed problem requires

* Both authors contributed equally.

Fig. 1. Our approach models a three-dimensional object in terms of a graphical model, where the nodes correspond to 3D part positions and the edges encode geometric constraints between part positions. Given an input image and a viewpoint estimate the cost function of our energy model dictates the optimal placement of the parts in *three dimensional space*, obtaining a reconstruction of the object's surface from a monocular image.

somehow introducing knowledge about the 3D geometry of an object through a model-based approach.

A model-based approach to solving such a problem typically involves (a) using multiple images during training to construct a three-dimensional surface model, and (b) interpreting a single image at test time by adapting the model to it. It is clear that restricting the surface reconstruction task to a specific category simplifies the generic surface reconstruction problem [14,1,21], since we now have a more specific, model-based prior knowledge about the anticipated solution. The main question is which model is best suited for this task.

Our approach builds on the recent advances of [17] where it was shown that accurate surface reconstructions can be obtained from RGB images for a wide variety of categories in the PASCAL VOC dataset. We use the same procedure for recovering the 3D geometry of categories from 2D datasets, but change entirely their modelling approach, which leads to different learning and optimization tasks. The work of [17] was using iterative optimization algorithms that can get stuck at local minima, while the cost function driving the optimization was hand-crafted, making it hard to profit from rich features or large datasets. We propose instead an approach that comes with guarantees of obtaining a globally optimal solution, and develop an associated Structured SVM training algorithm that can optimally design the cost function for the task at hand.

For this, we introduce a graphical model inspired from the Deformable Part Model paradigm [10,8], treating part positions as nodes of a graph and incorporating geometrical constraints within graph cliques. In particular, we lift Deformable Part Models [8] to 3D, allowing the object parts to live in a 3D pose space, reflecting the 'real-world' part coordinates. We represent prior information about the 3D shape of an object category in terms of viewpoint-conditioned pairwise terms among parts, and use rich CNN features to construct unary terms that dictate the part placement in the image. In order to make this three-dimensional model practically exploitable we develop customized optimization and learning techniques.

Regarding optimization, our method is guaranteed to deliver the optimal solution when using a loop-free graph, like a star or a tree. Even though inference

would in principle be possible using generic efficient message-passing algorithms, such as Generalized Distance Transforms [9], the 3D nature of our task makes such techniques memory- and time- inefficient: the complexity scales linearly in the depth resolution, so one would need to trade off accuracy for speed. Instead, in Sec. 4 we extend Dual-Tree Branch-and-Bound [19] to 3D, keeping the memory complexity constant and the computational complexity logarithmic in the depth resolution - this allows us to use a fine-grained depth resolution with negligible computational overhead.

Regarding learning, we use discriminative training which allows us to design the cost function so as to place the model predictions as close as possible to the ground-truth surface. In particular we use Structured SVM training with a loss function that penalizes the deviation between the estimated 3D position of the parts and the ground-truth position of the associated object landmarks. We jointly learn how to score the CNN features that are used for the construction of our unary terms and the displacement features that are used for our pairwise terms, while it would be also possible to backpropagate on the CNN that delivers the CNN features.

We demonstrate the merit of our contributions through systematic comparisons on the PASCAL 3D+ dataset, obtaining better results than the current state-of-the-art method of [17] on most categories.

2 Related Work

There are several problems pertaining to our task, including (i) the acquisition of 3D geometry from RGB images, (ii) the modelling of 3D deformations, and (iii) the interpretation of a test image in terms of a learned 3D deformation model. We start by presenting the techniques underlying the current state-of-the-art method of [17] and then turn to techniques that are closer to our own work.

Regarding (i), several techniques have recently tackled the problem of establishing 3D geometric models from RGB images [5,16,18], while a closely related task aims at establishing non-rigid correspondences across unstructured sets of images [4,29]. We follow the work of [26,17] where a minimal annotation, in terms of a few landmarks per object instance, is combined with non-rigid structure from motion algorithms to lift the PASCAL dataset to 3D and estimate surface models for 10 categories.

Turning to (ii), the modeling of shape variability, the deformable model paradigm used in [17] relies on a linear, low-dimensional subspace to parameterize the possible surface variation that a category can have with respect to the mean, nominal surface. A particular shape instance can thus be expressed in terms of a low-dimensional coefficient vector.

Turning to (iii), adapting a deformable shape model to a novel image, the authors of [17] estimate the coefficients of the surface model so that the projection of the estimated 3D surface on the 2D camera plane is aligned with the object silhouette, while some designated mesh points project close to two-dimensional landmark positions; segmentation and landmarks are provided either by humans

or by external modules [17]. While exploiting several acceleration techniques that are particular to the cost function being used, the optimization time is in the order of seconds and the gradient descent procedure used by the authors is prone to getting stuck in local minima.

Turning to works that rely on DPMs, as we do, several works have recently aimed at coupling DPMs with three-dimensional modelling, e.g. [22,23,28,30,31]. However, none of the existing works treats the 3D positions of the parts as variables that are being optimized over. We note that in our case we have $M = 100$ landmarks in $3D$ while in most of these works the three-dimensional variable being optimized corresponds to the relative angle of the camera to the object [22,23,28,30,31,13] and potentially also the visibility of points [30,28]. Several works go beyond this and search also over CAD models, typically represented as exemplars, e.g. [22], while certain other works, e.g. [23,30], also search over part positions in 2D.

In our case we actually consider that the viewpoint is provided by an external module or by ground-truth annotations. Our modelling part is fairly similar to that of [23], but the major difference lies in the solution space: to the best of our knowledge we are the first work that achieves efficient and guaranteed optimal placement of object parts in 3D. Our focus is on the solution of the combinatorial optimization problem obtained by trying to jointly position all 3D parts of an object in a mutually consistent configuration. For this we have built on fast techniques for inference with 2D DPMs, [19,2], and adapted them to operate in 3D.

3 3D Deformable Part Models for Category Surface Estimation

Given an image I and a viewpoint v, our task is to estimate the 3D coordinates of P mesh nodes $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_P\}$, where every node is a 3D position vector $\mathbf{x}_i = (h_i, v_i, d_i)$. We will be denoting vectors with boldface letters and will alternate between the vector notation $\mathbf{x}$ and the horizontal/vertical/depth notation (h, v, d) based on convenience.

For this we use Deformable Part Models [8] (DPMs) which provide a rigorous, energy-based approach to modelling and detecting deformable object categories. As we describe below, DPMs can also be adapted to our problem. We break the problem into three steps; firstly in Sec. 3.1 we determine the 3D model, defining the graph nodes and relationships between them. Secondly, in Sec. 3.2 we determine how the surface would look like when projected in 2D. Finally, in Sec. 3.3 we describe how to use the image so as to indicate which of the possible 3D surfaces projects in a way that matches with the image observations.

3.1 Model graph construction

We start by describing how we obtain the nodes of the DPM graph, which requires having access to 3D geometry related to an object category. As in [17], we rely on

the work of [26] to recover 3D geometry from a 2D dataset, such as the PASCAL VOC dataset; given a set of images of an object category and K category-specific keypoint annotations, Non-Rigid Structure-from-Motion (NRSfM)[3] delivers a joint estimation of (1) the 3D position of the K category-specific keypoints (including occluded keypoints) and (2) a camera viewpoint estimate for each training instance. Given the viewpoint estimates, a class-specific triangulated 3D mesh is obtained from ground-truth segmentation masks by constructing a visual hull from the training instance.

This 3D mesh and the K 3D-lifted keypoints are associated by projecting every keypoint to the 3D mesh point that is closest to it on average over the training set. This provides us with the first K nodes of our model. A denser sampling of the mesh allows our model to more thoroughly capture the object's shape; we obtain $P = 100$ nodes using Geodesic Surface Remeshing [24], which ensures that the K keypoints are selected, while the remaining points are roughly equidistant. In Fig. 1(a) we illustrate an example of the input and the downsampled mesh for the car category.

We obtain a Deformable Part Model by treating each of those mesh points as a graph node. Edges between nodes capture geometric constraints; each edge $e_{i,j}$ is associated with a 3D nominal displacement vector $\mu_{i,j}$ equalling the average displacement between nodes i and j, and a spherical precision matrix $C_{i,j}$ indicating the typical inverse variance around that displacement. We note that the edges do not coincide with the ones in the mesh triangulation described above. In particular, we use edges that connect potentially opposite surface points, as these may better capture volume constraints.

3.2 Viewpoint-adapted 3D DPMs

Having defined the model's 3D geometry, we now turn to accounting for the effects of camera pose, so as to connect our model to the 2D images that will be available at test time. This includes 6 degrees of freedom, 3 for translation, and 3 for rotation.

We consider momentarily that the object is at a fixed depth, which amounts to working with images that are scale-normalized, where scale is indicated either by an object detection module, or by a ground-truth bounding box. As our experiments show, searching around this originally estimated scale can yield some improvements, but we ignore it from the following discussion. Regarding horizontal and vertical translation, our algorithm effectively does an efficient, exhaustive search.

The remaining degrees of variation include the camera's azimuth, elevation and rotation with respect to the object's canonical coordinate system in 3D. For this we introduce an additional viewpoint variable, v, which can either be provided by the output of NRSfM (which requires multiple images and landmark annotations), or by a bottom-up viewpoint estimator, as e.g. in [25]. In all of our experiments we use the NRSfM-based viewpoint estimate, and compare to results of systems that use the same viewpoint estimate. One can actually search

for the viewpoint through the optimization of our model's objective, but we leave this for future work.

This viewpoint variable influences the model in two ways. Firstly, the 3D nominal displacements are rotated in 3D by multiplying with R_v, the rotation matrix corresponding to v; we denote this by adding a viewpoint superscript to the nominal displacement vector: $\mu^v_{i,j} \doteq R_v \mu_{i,j}$. The precision matrix, being spherical, is unaffected by viewpoint; dealing with elliptical matrices is equally easy, but the induced coupling of coordinates would pose challenges when we turn to optimization in Sec. 4. Secondly, the variable v is discretized into V distinct viewpoints, obtained by uniformly quantizing the 360 azimuth degrees; we denote by $\lfloor v \rfloor$ the discretized value. This is used to determine in a viewpoint-dependent manner the image-to-part affinities, as detailed below.

3.3 Cost Function

Having identified the model variables and the 3D-to-2D transformation, we can now describe how we go from 2D to 3D, when presented with an image. The image provides us with 'bottom-up' evidence about the 2D positions where each visible landmark is likely to be seen. This expressed in terms of a landmark-specific unary term of the form:

$$\mathcal{U}_{I,v,i}(\mathbf{x}_i) = \langle \mathbf{u}_i^{\lfloor v \rfloor}, f_I(h_i, v_i) \rangle, \tag{1}$$

where $\mathbf{u}_i^{\lfloor v \rfloor}$ is a viewpoint-specific weight vector for node i and f_I are dense image features. We use orthographic projection, assuming that the 3D landmark $\mathbf{x}_i = (h_i, v_i, d_i)$ projects to the 2D image point (h_i, v_i). We note also that the argument of the unary term, $\mathbf{x}_i$ is three-dimensional, while the underlying information is two-dimensional.

The geometric arrangement between two landmarks i, j is controlled by a function of two 3D arguments that prescribes preferences for the arrangement of landmark pairs in 3D:

$$\mathcal{V}_{i,j,v}(\mathbf{x}_i, \mathbf{x}_j) = -\left(\mathbf{x}_j - \mathbf{x}_i - \mu^v_{i,j}\right)^T C_{i,j} \left(\mathbf{x}_j - \mathbf{x}_i - \mu^v_{i,j}\right), (i,j) \in \mathcal{E}. \tag{2}$$

In Eq. 2 $\mu^v_{i,j} = R_v \mu_{i,j}$ is the viewpoint-adapted nominal displacement and $C_{i,j}$ is the viewpoint-invariant precision matrix described in the previous subsection. Since $C_{i,j}$ is a spherical precision matrix, we can write $\mathcal{V}_{i,j,v}(\mathbf{x}_i, \mathbf{x}_j) = \gamma_{i,j} \|\mathbf{x}_j - \mathbf{x}_i - \mu^v_{i,j}\|^2$, showing that our model penalizes the ℓ_2 norm of the deviations of the 3D part displacements from their rotated nominal value.

Putting things together, given an image I and a viewpoint v, we score a landmark configuration $\mathbf{X}$ with a merit function S formed as the sum of unary and pairwise terms:

$$S_{I,v}(\mathbf{X}) = \sum_{i=1}^{P} \mathcal{U}_{I,\lfloor v \rfloor,i}(\mathbf{x}_i) + \sum_{(i,j)\in\mathcal{E}} \mathcal{V}_{i,j,v}(\mathbf{x}_i, \mathbf{x}_j), \tag{3}$$

where $\mathcal{E}$ is the set of edges on the graph. The unary terms introduce image-based evidence, and the pairwise terms enforce model-based priors. The minimization of this objective will result in an optimal placement of K landmarks in 3D while having a 2D input.

4 Efficient Optimization for 3D DPMs

Having developed our cost function in 3D we now turn to its optimization. We consider a star-shaped graph, where a single node ('root') is connected to all remaining nodes ('leaves'). The logarithm of the root node's max-marginals for a star-shaped graph equal:

$$S(\mathbf{x}_r) = \max_{\mathbf{X}:\mathbf{X}_r=\mathbf{x}_r} S(\mathbf{X}) = \sum_i \max_{\mathbf{x}_i} \left[\mathcal{U}_i(\mathbf{x}_i) + \mathcal{V}_{i,r}(\mathbf{x}_i, \mathbf{x}_r)\right], \tag{4}$$

effectively scoring a candidate root position $\mathbf{x}_r$ by optimizing over all possible part positions that may support it. In order to simplify notation we have removed the image and viewpoint dependence and introduced $\mathcal{V}_{i,r}(\mathbf{x}_i, \mathbf{x}_r)$ which is defined to be zero when $\mathbf{x}_i = \mathbf{x}_r$ and infinity otherwise.

For the particular form of the pairwise term used here one can use the Generalized Distance Transform (GDT) of [9], reducing the computational complexity of message-passing from $O(N^2)$ to $O(N)$ in the number of voxels. This can be fast in 2D, but in 3D we face a linear increase of complexity and memory in the depth resolution - so we will need to eventually tradeoff speed for accuracy.

However, we realize that the Dual-Tree Branch-and-Bound (DTBB) algorithm of [19] directly applies to this problem. In particular, in DTBB rather than first exhaustively computing Eq. 4 for all values of $\mathbf{x}_r$ and then picking the maximum, one instead performs prioritized search for the maximum - which can replace a complexity of $O(N)$ with one of $\omega(\log N)$, i.e. logarithmic in the best-case, which is typically orders of magnitude faster. Furthermore, since for our case the unary terms are depth-independent, one never needs to occupy memory to represent $U_r(\mathbf{x}_r)$, as would be required by GDT - instead our algorithm's memory complexity is *constant* in depth resolution.

In a bit more detail, the DTBB algorithm performs prioritized search over intervals of root positions, denoted by X, using as priority an upper bound to the score in the interval. For this one bounds the right side of Eq. 4 over a root interval X, using the following series of inequalities:

$$\begin{aligned} \max_{\mathbf{x}_r \in X} S(\mathbf{x}_r) &\leq \sum_i \max_{\mathbf{x}_r \in X, \mathbf{x}_i} \mathcal{U}_i(\mathbf{x}_i) + \mathcal{V}_{i,r}(\mathbf{x}_i, \mathbf{x}_r) \\ &\leq \sum_i \max_{\mathbf{x}_i} \mathcal{U}_i(\mathbf{x}_i) + \sum_i \max_{\mathbf{x}_r \in X, \mathbf{x}_i} \mathcal{V}_{i,r}(\mathbf{x}_i, \mathbf{x}_r), \end{aligned} \tag{5}$$

where both inequalities follow from $\max_x f(x) + g(x) \leq \max_x f(x) + \max_x g(x)$. Intuitively, the first inequality opportunistically uses contributions from different nodes, even if they do not agree on the particular $\mathbf{x}_r \in X$ that they support, and

the second inequality opportunistically uses contributions from the unary and pairwise terms of a single node, even if they do not agree on the value of $\mathbf{x}_i$ that lends the support.

Two facts prove useful in computing Eq. 5 efficiently: firstly, the maximization $\max_{\mathbf{x}_i} U_i(\mathbf{x}_i)$ can be computed in 2D, since $U_i(\mathbf{x}_i)$ is independent of the d_i component of $\mathbf{x}_i$. As such, we never need to explicitly allocate memory to store the unary term in 3D, as would be required by GDTs. Secondly, the maximization over the combinations of $\mathbf{x}_i, \mathbf{x}_r$ can be computed *analytically*, with a cost that is constant, and independent of the cardinality of the two sets; all one needs to do is extend the 2D bounding schemes of [20] to 3D, and the complexity increases from 4 summation and multiplication operations (in [20]) to 6. Other than these two observations, we use the exact same algorithm as in [19,20]; for lack of space we refer the interested reader to those references for a more thorough presentation of DTBB.

Another substantial accelaration can be obtained by noting that the pairwise term is invariant to depth translations. Combining this with the depth-independent nature of our unary terms, means that we have a one-dimensional family of equally good solutions. Intuitively, the only constraint imposed by our model consists in properly 'unfolding' the object in 3D space, while the particular depth value around which this happens is irrelevant. As illustrated in our experimental results in 2, Branch-and-Bound would waste time on exploring evenly all those equivalent solutions; instead we fix the model's root node at a fixed depth, and use it as an anchor point for the leaf nodes.

5 Model learning

Having outlined our cost function and our optimization algorithm we now turn to parameter estimation. Given a candidate configuration $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_P)$ of P parts, we treat our cost function in 3 as the inner product between an image-specific feature vector $\mathbf{F}(\mathbf{X})$ and a model-specific weight vector $\mathbf{w}$, $S(\mathbf{X}) = \langle \mathbf{w}, \mathbf{F}(\mathbf{X}) \rangle$.

The feature vector comprises the P D-dimensional vectors extracted from the part positions $\mathbf{f}(\mathbf{x}_p), p = 1, \ldots, P$, and the model deformations with respect to the nominal displacements from the leaves to the root node $(x_p - x_l - \mu^v_{r,p})^2, p = 2, \ldots, P$ for each of the 3 dimensions. This accounts for a total size of $PD+3(P-1)$, which is also the dimension of the learned weight vector. We do not estimate $\mu^v_{r,p}$ discriminatively, as this would impede the viewpoint-based coupling of these parameters described in Sec. 3.2. We do however estimate the relative weight of the above ℓ_2 distance and use the average displacement to define μ, which combined with the viewpoint v yields $\mu^v_{r,p}$. The unary terms of graph nodes that do not correspond to one of the human-annotated keypoints are set to 0, meaning that the image evidence does not affect their position - while for a given viewpoint we set to zero the features of points that are not visible.

In order to learn the corresponding parameter weights, we use Structured Support Vector Machine (SSVM) training a cutting-plane optimizer [15], as

done in the context of Deformable Part Models e.g. in [23,2]. In particular we measure the performance of a particular weight vector in terms of a loss function $\Delta(\mathbf{X}^*_{I_i}, \hat{\mathbf{X}}_i)$ which represents the cost incurred by labelling image i as $\mathbf{X}^*_{I_i}$ when the ground truth is $\hat{\mathbf{X}}_i$. We choose to use the Mean Euclidean Distance, $\Delta(\hat{\mathbf{X}}, \mathbf{X}) = \frac{1}{P}\sum_{p=1}^{P} \|\mathbf{x}_p - \hat{\mathbf{x}}_p\|_2$ as a loss for our learning task, penalizing the 3D displacement of our estimated landmarks from their ground truth positions. This loss decomposes over each node of the graph. This allows us to leverage the tractability of the inference in the learning procedure, as is common in Structured SVMs.

More specifically, each iteration of the cutting plane algorithm requires the computation of the most violated constraint given the current estimate of the joint weight vector w^i, finding a configuration $\mathbf{X}^i_{cp}$ that is consistent with the model, yet incurs a high loss. This subroutine corresponds to the optimization problem:

$$\mathbf{X}^i_{cp} = \arg\max_{\hat{\mathbf{X}}} S_{I,v}(\hat{\mathbf{X}}) + \Delta(\hat{\mathbf{X}}, \mathbf{X}) \tag{6}$$

for each instance $(I, \mathbf{X})$ of the labelled training set. By incorporating the loss in the expression of our model's configuration score , this problem is cast as an equivalent inference problem with modified unaries, scoring a configuration as:

$$\tilde{S}_{I,v}(\hat{\mathbf{X}}) = \sum_{i=1}^{P} \left(\mathcal{U}_{I,\lfloor v \rfloor,i}(\mathbf{x}_i) + \delta(\hat{\mathbf{x}}_i, \mathbf{x}_i)\right) + \sum_{(i,j)\in\mathcal{E}} \mathcal{V}_{i,j,v}(\mathbf{x}_i, \mathbf{x}_j) \tag{7}$$

where $\delta(\hat{\mathbf{x}}_i, \mathbf{x}_i) = 1/P\|\hat{\mathbf{x}}_i - \mathbf{x}_i\|$. The complexity of the subroutine is therefore equivalent to the complexity of our inference algorithm. This framework allows to learn the joint weight vectors in a tractable time of the order of a few hours per view and category.

6 Results

We use the Pascal 3D+ dataset for all of our experiments [27], comprising 1408 images over 10 object categories. The ground truth 3D objects are given by the dataset via rotated and positioned CAD models. We use these CAD models solely for the evaluation of our method.

For feature computation we extract dense image features using the Deeplab network of [6], which is a fully convolutional neural network (FCNN) trained for semantic segmentation. We use the intermediate-level layer activations from layers 3 and 5 yielding a position-dependent feature vector of dimension $D = 769$. Using both lower- and mid-level features gives our unary terms the option of achieve both good localization results (low-level), and good invariance to intra-class appearance variability (high-level).

We evaluate our models in terms of inference acceleration and model performance. Firstly we benchmark the runtime of Branch-and-Bound in 3D and

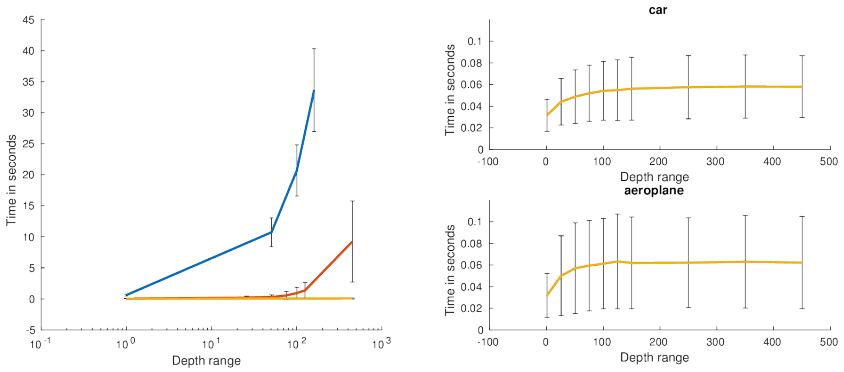

Fig. 2. Left: acceleration of branch-and-bound versus Generalized Distance Transforms as a function of depth resolution. The yellow curve shows the runtime of inference with our Branch-and-Bound algorithm in 3D. The red curve reveals the effect of not anchoring the model in depth, whereas the blue curve represents the implementation with GDTs. Branch-and-Bound is orders of magnitude faster than the GDT based implementation even on moderate depth ranges. Right: Branch-and-Bound performance on two indicative categories, car and aeroplane. The runtime of our algorithm is hardly affected when increasing the number of depth layers.

compare it to message-passing with Generalized Distance Transforms [9], commonly used in Deformable Part Models. Secondly we demonstrate that our method outperform the previous state of the art of [17] on 7 out of 10 categories using multi-scale of the Pascal 3D+ dataset. The single-scale version doesn't decisively outperform [17] (5/10), but yields comparable results in much faster runtime.

6.1 Branch-and-Bound in 3D

To benchmark the acceleration delivered by Branch-and-Bound in 3D we test our algorithm with a varying number of depth layers n_d; Our algorithm has a logarithmic best-case complexity, while inference with the Generalized Distance Transform (GDT) ([9]) scales linearly in n_d.

Figure 2 on the left empirically demonstrates that this is the case, showing the runtime of Branch-and-Bound on a varying number of depth layers (yellow graph). We set it against a Generalized Distance Transform (GDT) based implementation (blue graph). The timings have been obtained by averaging over the individual contributions of 100 images, 10 from each category, the average standard deviation has been annotated in the graph. The GDT-based inference is about two orders of magnitude slower than the inference with Branch-and-Bound if we set $n_d = 100$. The red graph displays the runtime of Branch-and-Bound when the model is not anchored in depth to distinguish between otherwise equivalent solutions.We note that above a certain size of depth range the computation times increases sharply, indicating that the number of equivalent solutions increases and Branch-and-

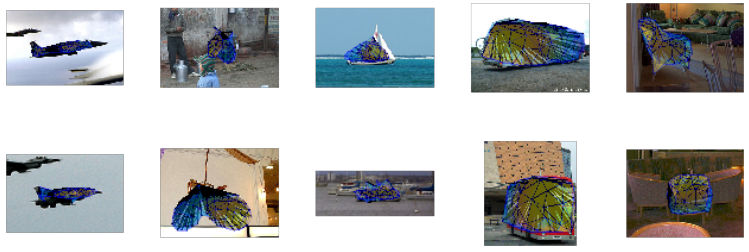

Fig. 3. Some reconstructed surfaces in 5 categories.

Bound explores these solutions. By contrast the anchored Branch-and-Bound only increases logarithmically in n_d, being able to perform inference in less than 0.2 seconds.

6.2 Surface Estimation

Following the methodology of Kar *et al.* [17], we measure the performance of our method by using the Hausdorff distance d_H to compare the system's output with the ground truth, $d_H(X,Y) = \max\{\sup_{x\in X} \inf_{y\in Y} d(x,y), \sup_{y\in Y} \inf_{x\in X} d(x,y)\}$, with points $x \in X$ being the points on the surface of the ground truth shape X and points $y \in Y$ being the points on the surface of the inferred shape Y following rigid alignment.

We display in Table 1 the average Hausdorff distance between the inferred mesh of our star-graph model, and reference CAD model associated to the object instance. In the third row of this table, we show that applying the model on different scales improves the Hausdorff error, which hints at possible future improvements of our method incorporating viewpoint estimation in the inference. Moreover, Fig. 3 shows some examples of inferred placed models. We observe that despite the simplicity of star-shaped graphical models and the speed of the inference, we are competitive to [17], obtaining a better accuracy [17] in most categories.

Table 1. Mesh errors computed using the Hausdorff error with respect to the centered ground-truth CAD models provided in Pascal VOC 3D+ (lower is better).

	plane	**bike**	**boat**	**bus**	**car**	**chair**	**sofa**	**train**	**tv**	**mbike**	**mean**
Kar *et al.* [17]	**2.2**	4.4	6.0	3.9	3.2	**2.6**	8.8	6.6	4.5	**2.9**	4.5
1 star	2.4	4.1	6.1	4.1	**3.1**	3.2	**5.3**	6.3	3.4	3.0	4.1
1 star multiscale	2.4	**4.0**	**5.7**	**3.8**	**3.1**	3.0	5.5	**6.0**	**3.3**	3.0	**4.0**

7 Conclusion and Future Work

In this work we have shown that Branch-and-Bound is suitable to infer DPMs in 3D and can preserve its logarithmic scalability. To this end we also provide a method to learn DPMs from silhouette information and viewpoint annotations. The model is simple, but outperforms the model of Kar et a. [17] in some categories by a significant margin. On average over categories we yield an 11% smaller Hausdorff error.

From our results, we see that BB can be a good option for rapid inference. This shows that future work is needed to explore the full potential of BB. Our future investigations will also aim at extending Branch-and-Bound to more general energies (for example loopy graph structures) and other types of potentials.

Using an energy-based model also leaves open the potential of coupling structured prediction techniques with CNN training [7] for the task of surface estimation. We leave this last direction for future work, but we consider it as a promising direction that is opened by our energy-based approach.

8 Acknowledgement

This research has been supported by the FP7-RECONFIG and H2020–I-SUPPORT projects of the European Union.

References

1. J. T. Barron and J. Malik. Shape, illumination, and reflectance from shading. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(8):1670–1687, 2015.
2. H. Boussaid and I. Kokkinos. Fast and exact: Admm-based discriminative shape segmentation with loopy part models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4058–4065, 2014.
3. C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3d shape from image streams. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 690–696. IEEE, 2000.
4. J. Carreira, A. Kar, S. Tulsiani, and J. Malik. Virtual view networks for object reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 2937–2946, 2015.
5. T. J. Cashman and A. W. Fitzgibbon. What shape are dolphins? building 3d morphable models from 2d images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(1):232–244, 2013.
6. L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.
7. L.-C. Chen, A. G. Schwing, A. L. Yuille, and R. Urtasun. Learning Deep Structured Models. In *Proc. ICML*, 2015.
8. P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

9. P. F. Felzenszwalb and D. P. Huttenlocher. Distance transforms of sampled functions. Technical report, Cornell CS, 2004.
10. M. Fischler and R. Erschlanger. The Representation and Matching of Pictorial Structures. *IEEE Trans. Comput.*, 22(1):67–92, 1973.
11. R. B. Girshick, F. N. Iandola, T. Darrell, and J. Malik. Deformable part models are convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 437–446, 2015.
12. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
13. M. Hejrati and D. Ramanan. Analysis by synthesis: 3d object recognition by object reconstruction. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2449–2456. IEEE, 2014.
14. B. K. P. Horn and M. J. Brooks, editors. *Shape from Shading*. MIT Press, Cambridge, MA, USA, 1989.
15. T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.
16. Kanazawa, D. Jacobs, and M. Chandraker. WarpNet: Weakly Supervised Matching for Single-View Reconstruction. In *CVPR*, 2016.
17. A. Kar, S. Tulsiani, J. Carreira, and J. Malik. Category-specific object reconstruction from a single image. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 1966–1974. IEEE, 2015.
18. I. Kemelmacher-Shlizerman. Internet based morphable model. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 3256–3263, 2013.
19. I. Kokkinos. Rapid deformable object detection using dual-tree branch-and-bound. In *NIPS*, 2011.
20. I. Kokkinos. Bounding Part Scores for Rapid Detection with Deformable Part Models. In *2nd Parts and Attributes Workshop, ECCV*, 2012.
21. T. D. Kulkarni, P. Kohli, J. B. Tenenbaum, and V. K. Mansinghka. Picture: A probabilistic programming language for scene perception. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 4390–4399, 2015.
22. J. J. Lim, A. Khosla, and A. Torralba. Fpm: Fine pose parts-based model with 3d cad models. In *Computer Vision–ECCV 2014*, pages 478–493. Springer, 2014.
23. B. Pepik, M. Stark, P. V. Gehler, and B. Schiele. Multi-view and 3d deformable part models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(11):2232–2245, 2015.
24. G. Peyré and L. D. Cohen. Geodesic remeshing using front propagation. *International Journal of Computer Vision*, 69(1):145–156, 2006.
25. S. Tulsiani and J. Malik. Viewpoints and keypoints. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1510–1519, 2015.
26. S. Vicente, J. Carreira, L. de Agapito, and J. Batista. Reconstructing PASCAL VOC. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 41–48, 2014.
27. Y. Xiang, R. Mottaghi, and S. Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014.
28. E. Yoruk and R. Vidal. Efficient object localization and pose estimation with 3d wireframe models. In *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on*, pages 538–545. IEEE, 2013.

29. T. Zhou, Y. J. Lee, S. X. Yu, and A. A. Efros. Flowweb: Joint image set alignment by weaving consistent, pixel-wise correspondences. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1191–1200, 2015.
30. M. Zhu, X. Zhou, and K. Daniilidis. Single image pop-up from discriminatively learned parts. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 927–935, 2015.
31. M. Z. Zia, M. Stark, and K. Schindler. Are cars just 3d boxes? jointly estimating the 3d shape of multiple objects. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 3678–3685, 2014.